January 11, 2007

start:FPGAs vs. DSPs: A look at the unanswered questions

BDTI looks at the open questions about FPGAs' performance, cost, power, and ease of development. It also explains why FPGAs might benefit from the move to deep-submicron processes.

CONTENT PROVIDED BY: BDTi

BDTI recently completed an in-depth analysis of FPGAs' suitability for DSP applications. We found that, in some high-performance signal processing applications, FPGAs have several significant advantages over high-end DSP processors. Our recent benchmark results (shown in *Figure 1*), for example, have shown that high-end, DSP-oriented FPGAs have a huge throughput advantage over high-performance DSP processors for certain types of signal processing. And FPGAs, which are not constrained by a specific instruction set or hardwired processing units, are much more flexible than processors.

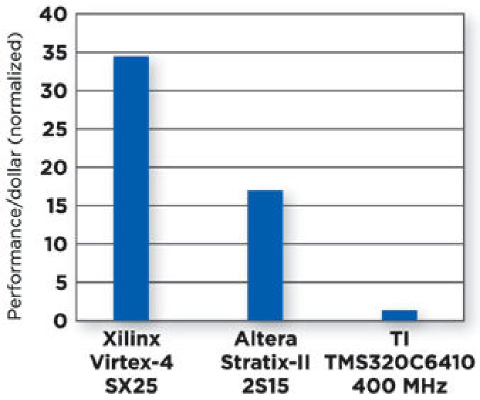


Figure 1. Results of the BDTI Communications Benchmark (OFDM)™

If market success were based solely on throughput or flexibility, FPGAs would appear to be on the verge of taking over the DSP market; in fact, according to a recent report from market research firm Forward Concepts, in 2005 Altera and Xilinx each had DSP FPGA revenues in excess of $200 million, selling more non-cell-phone DSP silicon than Freescale and Agere.

But of course, it's not that simple. Development effort, energy efficiency, cost-effectiveness, staff expertise, and market inertia (among other attributes) will all play a role in determining whether FPGAs become a dominant technology for DSP systems.

In this article, we'll share some of the key open questions that we've identified during the course of our analysis. These factors will affect FPGAs' success in DSP markets, and will be of significant interest to system designers who are considering using FPGAs in their signal processing systems.

Are FPGAs energy hogs, or not?

Energy efficiency is often a critical metric for signal processing applications.

Battery-powered products are highly sensitive to energy consumption, and even line-powered products are often sensitive to power consumption, though it may be on a per-channel or per-unit-area basis. FPGAs have long been viewed as too power-hungry for most DSP applications, but we believe that this may be an obsolete perspective.

FPGAs use highly flexible architectures, and this flexibility is perhaps their greatest advantage. But flexibility comes with a hardware cost. More flexibility generally means more gates, more silicon area, more routing resources—and higher energy consumption. For this reason, FPGAs are generally less energy efficient than chips with dedicated hardware, such as ASICs.

But how do FPGAs compare to DSP processors? DSPs are highly tailored for efficient implementation of common DSP tasks, and thus many engineers assume that they are more energy-efficient than FPGAs. But DSP processors have their own inefficiencies. In a DSP, only a tiny fraction of the silicon is devoted to computation; most of the silicon area and most of the energy is devoted to moving instructions and data around. Hence, it would be a mistake to assume that FPGAs are inherently less energy efficient than DSPs.

In some high-performance signal processing applications, for example, FPGAs can take advantage of their highly parallel architectures and offer much higher throughput than DSPs. As a result, FPGAs' overall energy consumption may be significantly lower than that of DSP processors, in spite of the fact that their chip-level power consumption is often higher.

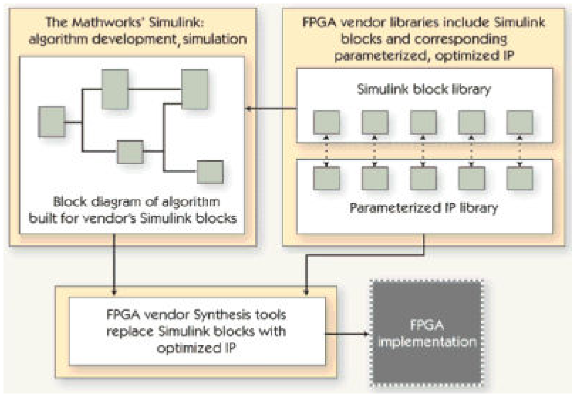
Unfortunately, there is a dearth of accurate, apples-to-apples energy consumption data for FPGAs and DSP processors, making it difficult to compare their energy efficiency As part of the analysis for our recent report comparing FPGAs to DSPs, "FPGAs for DSP, 2nd Edition," BDTI did its own "back of the envelope" comparisons of the energy efficiency of FPGAs and DSPs. Based on anecdotal data about FPGA power consumption, we estimated that high-end FPGAs implementing demanding DSP applications, such as that embodied in the BDTI Communications Benchmark (OFDM)™, consume on the order of 10 watts, while high-end DSPs consume roughly 2-3 watts. Our benchmark results have shown that high-end FPGAs can support roughly 10 to 100 times more channels on this benchmark than high-end DSPs, suggesting that their energy consumption per channel is significantly lower than that of DSPs. This contradicts the common view that FPGAs are energy hogs.

Obviously our comparison is based on very rough estimates. While we believe that FPGAs' energy efficiency is likely to be competitive—or even superior—to that of DSPs in many high-performance signal processing applications, this is still an open question. What's needed is a rigorous, well-controlled analysis of power consumption under comparable conditions.

Furthermore, the above analysis is only meaningful in the context of high-performance signal processing applications. Estimating the relative energy efficiency of FPGAs and processors for less-demanding signal processing applications would require a slightly different benchmarking approach and the evaluation of different chips (e.g., lower-power processors and smaller FPGAs); this is another area that's ripe for further investigation.

Just how useful are the new high-level tools?
Implementing a DSP application on an FPGA typically takes much more effort than implementing the same application on a DSP processor—it's not uncommon for an FPGA implementation to require five times as long as the equivalent DSP processor implementation. This huge difference translates into higher cost and slower time-to-market, which can be deal-breakers for many DSP-oriented products. For this reason, a number of vendors have

introduced high-level synthesis tools to help address the challenges of implementing signal processing applications on FPGAs. These tools enable the user to generate an FPGA implementation from a high-level representation, such as a Simulink block diagram, thus sidestepping the need to work with cumbersome hardware description languages. *Figure 2* illustrates this concept.


The Mathworks' Simulink: algorithm development, simulation
FPGA vendor libraries include Simulink blocks and corresponding parameterized, optimized IP
Simulink block library
Parameterized IP library
Block diagram of algorithm built for vendor's Simulink blocks
FPGA vendor Synthesis tools replace Simulink blocks with optimized IP
FPGA implementation

(Click to enlarge)
Figure 2. Typical high-level tool flow

Such tools aren't a new idea: ten years ago there was a similar wave of high-level synthesis tools targeting ASICs. Unfortunately, those tools never caught on. They promised more than they could deliver, and required engineers to do their work in new and unfamiliar ways—by using new languages, for example. The new synthesis tools aren't yet in widespread use; most engineers still "program" FPGAs the old-fashioned way, using hardware languages—but that is slowly changing. Whether the tools ultimately cause a major shift in how FPGAs are used (and by whom) will depend on a number of factors, including the following:

1. For tools that support designs expressed in C, is it standard C, or a variant of C that's not actually portable and is unfamiliar to most engineers? Can designers use their existing C algorithm representation, or will they effectively have to rewrite their code to use the tool?
2. What level of efficiency (in terms of the throughput and resource usage of the generated design) is sacrificed by using high-level synthesis rather than doing the work by hand with traditional, RTL-based design? And what level of efficiency (relative to traditional design) is really needed for the tool to be attractive?
3. What is the scope of applications that the synthesis tool is intended to handle? Is it geared mainly for signal processing, or can it also handle packet processing?
4. How much of a design can the tool implement? For example, tool vendors often show how efficient their tools are at generating FIR filters from high-level requirements. That's useful, but designers need to get data into and out of that filter, perhaps using chip I/O pins, buffering, bitstream parsing, or other steps. Does the tool handle the complete design?
5. What building blocks are provided? If an application relies on unique blocks not supplied by the vendor, what are the implications for productivity and quality of results?

Can FPGAs use extra gates better than processors?
As the industry moves to 65 nm processes, 45 nm processes, and beyond, the most obvious and reliable benefit is the ability to pack more circuits onto a given silicon die. Traditionally, processor designers have leveraged this additional circuitry to implement more-complex processors. These processors boost performance via sophisticated instruction sets and microarchitectures, including deep pipelines and multiple execution units. But such architectural techniques quickly reach a point of diminishing returns. For

example, adding execution units to a processor results in smaller and smaller performance and efficiency gains as the number of execution units increases. This is due to difficulties in finding and extracting suitable parallelism from applications (especially when the applications are expressed in inherently sequential languages, like C) and to bottlenecks elsewhere in the processor, such as in data memory bandwidth. For this reason, we don't routinely see, for example, processors with 16 or 32 execution units. As a result, in recent years processor designers have used most of their expanded gate budgets to incorporate more on-chip memory. Increased on-chip memory can boost performance and efficiency, but again it quickly reaches the point of diminishing returns.

Recognizing that it has reached the point of diminishing returns in scaling single-core processors for higher performance with advancing fabrication processes, the processor industry has turned its attention to multi-core processors. However, with limited exceptions, the development process for mapping an application onto a multi-core processor differs in important ways from that associated with single-core processors; tools and techniques for software development on multi-core processors are not nearly as rich, mature, and widely understood as those for single-core processors. This is the key obstacle to the rapid, widespread adoption of multi-core processors.

In contrast, because FPGAs use silicon in a relatively homogeneous way, it is easier for FPGA manufacturers to take advantage of increased circuit density to deliver more computation resources to the user. This translates into higher throughput for applications that can make use of the added parallelism, without changing the methods used by developers to map their applications onto the chips.

This is not to say that implementing applications on FPGAs is easier than implementing applications on multi-core processors. Rather, our point is that, as fabrication processes enable more circuitry on a chip, FPGAs can make good use of this added capacity without requiring a change in the design tools and methods utilized by FPGA users. In contrast, as processor designers switch from single-core to multi-core architectures, in most cases new application development techniques and tools will be required.

If, as we've speculated here, FPGAs are able to make better use of additional gates than processors do, this may give them a significant advantage over the long term. But the outcome will also depend on whether processor vendors are able to deliver tools that make multi-core software development faster and less painful.

Conclusions

BDTI plans to conduct further analysis of FPGA energy efficiency and high-level tools in the future, and we will continue to evaluate the signal processing capabilities of new FPGAs and processors.

DSP processors have been the dominant processing engines for many DSP applications for decades, but that may be changing. FPGAs clearly have many advantages relative to DSP processors for some high-performance applications; while they are unlikely to completely eliminate the need for DSPs, they may well invade—or even take over—many of the applications in which DSP processors are used today.

About BDTI

BDTI enables engineers, marketers, and managers to make confident technical and business decisions about technologies for signal processing applications. For more BDTI resources, see www.BDTI.com.